# Design and Implementation of Ad-Hoc Collaborative Proxying Scheme for Reducing Network Energy Waste

Khan, R., & Khan, S. U. (2017). Design and Implementation of Ad-Hoc Collaborative Proxying Scheme for Reducing Network Energy Waste. *Digital Communications and Networks*, *3*(2), 118-128. https://doi.org/10.1016/j.dcan.2016.11.001

**Published in:**
Digital Communications and Networks

**Document Version:**
Publisher's PDF, also known as Version of record

Digital Communications and Networks 3 (2017) 118–128

HOSTED BY

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.elsevier.com/locate/dcan

# Design and implementation of Ad-Hoc collaborative proxying scheme for reducing network energy waste☆

Rafiullah Khan[a,*], Sarmad Ullah Khan[b]

[a] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, ECIT/CSIT - Northern Ireland Science Park, Queen's Road, Queen's Island, Belfast BT3 9DT, United Kingdom

[b] Electrical Engineering Department, CECOS University of IT and Emerging Sciences, Phase-6, Hayatabad, Peshawar, Pakistan

ARTICLE INFO

*Keywords:*
Green networking
Energy efficiency
Smartphone
Home gateway proxy
Power measurement
Universal Plug & Play

ABSTRACT

Network devices are equipped with low power states but they are rarely activated due to their inability of maintaining network connectivity. Recently, Network Connectivity Proxy (NCP) concept has been proposed in literature as an effective mechanism to exploit full potential of low power features on network devices by impersonating their virtual presence. However, the NCP concept faces several open issues and challenges especially related to proxying of TCP connections and majority of daily used proprietary closed-source applications. This paper presents a new approach for reducing network energy waste through intelligent collaboration among daily used devices (e.g., desktop computers, laptops, tablets, smartphones, etc). It guarantees to run applications on only and only one user device that is under active use at that specific moment. To reduce energy waste and allow idle devices to sleep, our approach also takes benefit from a light-weight home gateway proxy with very basic practically realizable functionalities. The proposed system not just reduces energy waste of fixed devices but also enables mobile devices to significantly improve their battery life. Our developed software prototypes allow devices to autonomously and seamlessly collaborate with each other without requiring any configuration or user input. Further, this paper also presents the basic performance evaluation of developed prototypes in real networking environments.

## 1. Introduction

A significant portion of the energy consumed by network devices is usually wasted due to under-utilization or staying powered-up even when idle [1]. About 2% of the overall US electricity is consumed by computers according to the Environmental Protection Agency (EPA) [2]. Whereas, computers in office environment stay idle for more than 60% of the time but are still kept powered-up [3]. The main reason due to which people never switch off their computers is to stay 'Always Connected' for remote access, VoIP and Instant Messaging (IM) clients and other Internet based applications. The Advanced Configuration and Power Interface (ACPI) proposed several low power management states for computers but they are rarely used due to their inability to maintain always-connected status. Thus, huge energy savings are possible if computers can extensively use ACPI low power states without losing the always-connected status [4].

The Network Connectivity Proxy (NCP) concept is recently proposed in literature as an effective mechanism to reduce the network energy waste by allowing unattended idle devices to sleep without losing network presence [5,6]. The NCP concept ensures that devices sleep for the maximum possible time and it wakes them up only when their resources are required [7]. To achieve this objective, the NCP seamlessly impersonates link, network, transport and application layers presence on-behalf of sleeping devices [5,8,9]. Although the NCP concept is quite beneficial,its practical realization is very complex and challenging especially when concerning proxying of TCP connections and ever increasing number of Internet-based applications [10]. Different proxying techniques have been investigated in the literature, however, they are always limited to a specific protocol or open source application [11,12]. The two most common ways of proxying applications are application specific stubs [8] and freeze/resume features [5,13]. Application stub is the light-weight version of original application derived or re-written from its source code. But freeze/resume approach requires modification to application to include new features. Both application proxying strategies are not applicable for proprietary closed-source applications. It is important to note that majority of commonly used applications in day-to-day life are proprietary closed source. Till now, the NCP concept lacks the capability to proxy closed-source applications.

Peer review under responsibility of Chongqing University of Posts and Telecommunication.

* Corresponding author.
*E-mail addresses:* rafiullah.khan@qub.ac.uk (R. Khan), sarmad@cecos.edu.pk (S.U. Khan).

http://dx.doi.org/10.1016/j.dcan.2016.11.001
Received 24 May 2016; Received in revised form 4 September 2016; Accepted 8 November 2016
Available online 14 November 2016

Similar to desktop computers, the Internet-connected smartphones also became an important part of our daily life. Today, the smartphones are capable to run the same latest operating systems as available for the desktop computers including Microsoft Windows, Ubuntu, etc. Thus, the smartphones are able to offer all the same features and run the same applications as the desktop computers. The energy savings in the case of smartphone has no significant importance as it is already a very low power device. However, reducing energy waste can help to significantly improve the battery life [14]. Similar to computers, the applications on smartphone also demand always-connected status. Thus, the applications such as viber, skype, facebook messenger, whatsapp, vonage and other VoIP and IM applications periodically transmit/receive presence or heartbeat messages over Internet. These applications not only consume the mobile data but also use phone resources such as CPU, memory and hardware components such as WiFi, 3G/4G or some built-in sensors. In short, the smartphones battery life can be significantly improved if they run applications only when desirable.

In our daily life, multiple devices run the same applications concurrently (especially the smartphone always stay connected). In this paper, we present a new approach that can be easily realized practically to achieve all the benefits of NCP. Our proposed approach uses intelligent collaboration among daily used fixed (e.g., desktop computers) and mobile devices (e.g., smartphones, tablets, laptops). Instead of proxying, it guarantees to run applications on one and only one user device at any given instant. Thus, the applications run on either a smartphone or tablet or laptop or desktop PC or any other device currently under active use. For example, if the user is using a desktop computer, it will run all applications while other devices will reduce energy waste by stopping the applications. When the user is not using desktop computer, it can enter into sleep mode and a smartphone will maintain the presence of applications. Our proposed approach also uses a light weight Home Gateway Proxy (HGP) but with very basic practically realizable features. The HGP impersonates basic networking protocols (i.e., ARP, PING) on behalf of sleeping desktop computers and wake them up whenever necessary (e.g., remote access connection request is received). The proposed approach not only reduces the entire network energy waste but also helps in improving significantly the battery life of smartphones by running the applications only for short periods when desirable. Our main design goal in this paper is to achieve autonomous and seamless communication among different user devices without requiring any user input or configurations. For this purpose, our developed prototypes are based on the flexible and reliable Universal Plug & Play (UPnP) architecture. The UPnP architecture is suitable for any network device and guarantees zero configuration, auto-discovery and seamless communication among devices of interest. Finally, this paper also evaluates the performance of developed prototypes in real networking environments.

The rest of the paper is organized as follows: Section 2 presents a brief literature review. Section 3 motivates the proposed framework by addressing the issues and challenges in previous approaches. Section 4 presents detailed architectural design of our proposed system. Section 5 presents the light-weight HGP with very basic set of practically realizable features. Section 6 presents the implementation of software prototypes. Section 7 presents measurements and performance evaluation. Finally, Section 8 concludes the paper.

## 2. Background and related works

The NCP concept has recently been investigated by researchers in order to reduce the idle energy waste of desktop computers. The NCP impersonates link, network, transport and application layers presence on behalf of sleeping devices and wakes them up only when their resources are required. Although the NCP concept is quite beneficial, it still faces many open issues and challenges. Most of the literature works targeted some specific protocols or open source applications such as UPnP, Gnutella, Jabber clients and others [15,16,11,8].

The two main open issues in the NCP concept are proxying of TCP connections and proprietary closed-source applications. The initial attempt in preserving open TCP connections was based on introducing a new TCP header field ‘Connection Sleep’ [17]. This header field informs the remote peer about the power state transition of the device. Another approach is based on introducing a ‘shim’ layer between the application and socket interface [18]. This shim layer is responsible to freeze, start or stop a TCP session. However, both of the approaches require changes to standard TCP/IP operations. Another approach is based on using external Srelay SOCKS service [6] which relays every TCP connection between both peers. However, this approach faces scalability and performance issues as every connection is established through a third-party.

Agarwal et al. in [8,19] proposed the idea of using applications specific stubs for proxying of applications. However, writing stubs is a quite complicated process and is only possible for open source applications. A similar approach has been used for proxying of xChat and kaduChat application in [4]. Client devices use application-specific routines which continuously analyze the heartbeat messages of applications. Client devices transfer heartbeat information to NCP before sleeping. This approach is quite challenging if payloads are encrypted or varying in unpredictable fashion. Bolla et al. in [5,13] proposed a new approach using TCP session migration and introduced freeze and resume features in applications. The applications on client devices freeze their operations and migrate their TCP sessions to NCP before sleeping. Applications resume and TCP sessions migrate back from NCP when the client device wakes up. However, the approach is quite interesting but requires changes to applications to include freeze/resume features. The authors in [19] also proposed the idea of using virtual machines, where each virtual machine is responsible to maintain presence on behalf of a single sleeping device. However, this approach requires much memory and processing power which limit the scalability. Finally, to estimate the expected economic benefits, authors in [1,3] analyzed real network traffic traces for home and office environments and estimated the savings.

Similarly, for mobile devices such as smartphones, multiple complementary techniques have been proposed in order to improve their battery life. Most of the techniques are software based such as putting Wireless NIC (WNIC) in low power state during idle periods [20,21]. The authors in [22] proposed a stateful proxy called ‘scepter’ which is placed in the network infrastructure and reduces the number of control and data bits during transmission. Another approach is based on a transparent proxying for dynamic power aware scheduling [23]. The proxy maintains separate connection with both end peers and uses buffers to transmit data in bursts. Such approach for delay sensitive applications (such as youtube, Internet radio) and BitTorrent application has been addressed in [24,25]. A quite different approach of putting user interface instead of processor into sleep state has been proposed in [26]. It wakes up the user interface after processing (i.e., once the results are available). The concept similar to NCP has also been addressed for mobile device [14,27]. The authors have addressed challenges in design and implementation, however, the open issue of NCP still remains unresolved, i.e., proxying of TCP sessions and closed-source applications.

Many smartphone applications on Android store (such as Go Power Master, Battery Doctor, Juice Defender, DU Battery Saver, PowerSaver etc) implement some interesting strategies. Stopping background running applications is the most common strategy which is based on some pre-defined conditions such as screen timeout, screen lock, specified battery threshold or specific time periods. Another interesting approach is turning OFF Internet connectivity/hardware components (such as WiFi, 3G/4G, Bluetooth, etc) during undesirable periods. Once again, the Internet connectivity can be linked with screen timeout, battery threshold, screen lock, pre-specified time periods or periodically switching ON and OFF to keep the background running applica-

tions updated. Finally, CPU rate adaptation is a built-in feature in most of the smartphones. However, some custom applications link this feature also with screen timeout, battery threshold etc.

This paper presents an intelligent collaborative proxying scheme between daily used fixed and mobile network devices. The proposed approach not only overcomes the limitations of NCP but also offers equivalent or higher savings as that of the NCP for fixed network devices and much improved battery life for mobile devices.

## 3. Motivation: issues and challenges in NCP concept

The problem statement for NCP is quite obvious but difficult to achieve the objectives of proxying sleeping devices at all layers of TCP/IP stack. The NCP concept is basically proposed to allow unattended devices to sleep while impersonating the presence of applications running on them. In fact, the NCP concept is very useful in reducing network energy waste but now the smartphones which became an important part of our daily life can also play their role in achieving this objective. People are always equipped with an Internet connected smartphone regardless of work hours. Thus, why not use an energy saving strategy in which one user device maintains presence for his other devices and so on. This approach can be easily realized in legacy networking environments and can also easily address issues and challenges of the NCP concept.

The most challenging task in NCP concept is the proxying of ever increasing number of applications on behalf of sleeping devices. The NCP maintains applications presence by sending or responding to periodic application specific heartbeat messages. This requires the application source code to understand the nature of heartbeat messages which is only possible for open source applications [8]. Further, if the NCP is implemented at the local level, the developers need to have the understanding of different programming languages. There is currently no useful technique available that can be used for proxying of proprietary closed source applications such as skype, viber, MSN messenger, VoIP clients, etc. Our proposed approach solves this issue by not proxying applications but instead running them on any active user device e.g., smartphone, laptop, desktop PC, tablets, etc. Thus, all our system needs is to keep track of all user devices, their priorities and operational status and run applications on only and only one device that is currently under active use.

The second important challenge in the NCP concept is keeping/preserving TCP connections open on behalf of sleeping client devices and resume the same connection back when they wake up. Preserving the TCP connections is very important as they are associated with different applications. Currently this objective is achieved through complex techniques such as TCP connection splitting, TCP connection migrations, end-to-end TCP connection management etc [6,18,5]. Each technique has its own pros and cons in terms of scalability and performance. However, in our proposal there is no need to preserve open TCP connections.

The NCP concept requires tracking its clients mobility for correct functioning. If the NCP client is a mobile device, it may move from one network to another network during the sleep mode. Mobility tracking is very important for the NCP to correctly proxy applications and wake-up its client whenever desirable. Complex techniques can be adopted to keep the NCP updated about its clients mobility [27]. Fortunately, our proposed system does not require to keep track of the positions of client devices. Client devices periodically check with our system to know if they need to run applications or not.

Another main issue in NCP concept is its coverage which is limited only to local network [10,28]. To make easy the proxying of proprietary closed source applications through collaboration with the application developers, it is important to take the NCP out of LAN boundaries and standardize it. The issues related to IP addresses collision will appear if the NCP covers for more than one local network (different local networks may use same NAT and private IP address space). However, this issue can be partially addressed by making the NCP NAT-aware and using a Universally Unique IDentifier (UUID) for each device which enables the NCP to distinguish a device from others. However, still some issues will exist especially related to NAT and Firewall e.g., waking up a sleeping client device. This process becomes more and more complicated if the NCP clients are two, three or more levels behind the NAT (NAT behind NAT). The two way communication between NCP and its client devices will be also a challenging issue due to NAT and Firewall. Fortunately, our proposed system don't impersonate IP addresses, but just keeps track of all devices and their operational status related to a particular user. Further, one way communication (always initiated from inside the local network) is enough in our proposed system and hence, there is no need to create any security hole in the local network.

## 4. Intelligent collaborative proxying scheme

This section presents our proposed intelligent collaborative proxying scheme among user devices for not just reducing the entire network energy waste but also improving the battery life of mobile devices.

### 4.1. Overview

Today, we have many different choices for computing devices ranging from smartphone to desktop computer. Desktop computers got popularity in last decade as a computing device that can run many different applications ranging from entertainment to communication and research. At the beginning, computers were equipped with low memory of 64 MB and storage of upto 10 GB. Today, modern available computers have upto 16 GB memory and 8 TB storage. In recent years, smartphone also rapidly evolved from simple calling device to a high computing device. Today, modern smartphones are equipped with upto 2.5 GHz quad core processors, upto 2 GB of memory and 128 GB of storage. The smartphones are penetrating at very rapid rate in the world and their specifications are improving day by day. Today, the smartphones can run the modern operating systems which are initially available for desktop computers such as Microsoft Windows and Ubuntu. In simple words, a modern smartphone can do everything that a desktop computer or laptop can do.

The energy efficiency is primarily important not just for fixed but also for mobile network devices. A good example of daily used mobile device is a smartphone which becomes an important part of our daily life. The economic savings from the energy efficiency of smartphone will be negligible as it is already a low power device. However, energy efficiency can significantly improve its battery life. Most of the energy consumed by smartphone is due to the Internet based applications which not just utilize data connection but also utilize smartphone resources such as CPU, memory and hardware components, e.g., WiFi or 3G/4G or some other built-in sensors. Thus, an Internet-connected smartphone consumes battery 3–4 times faster than when it is not using Internet.

A good example of daily used fixed network device is a desktop computer. The desktop computers are usually left poweredup 24/7 in offices or at homes just for the sake of maintaining network connectivity. A user needs network connectivity either for remote access or for Internet-based applications. The remote access objective can be easily achieved by the design of a light-weight HGP running on the gateway device which simply wakes-up the sleeping device when a new connection request is received for it on a given transport protocol and port (e.g., TCP:22 for SSH, TCP:3389 or UDP:3389 for remote desktop protocol). The HGP is addressed in details in Section 5. However, maintaining the applications presence (especially closed-source) is quite complex and challenging job for the NCP.

An intelligent collaboration among user devices to control the applications running on them can provide significant benefits in terms of reducing network energy waste and improving the battery life of

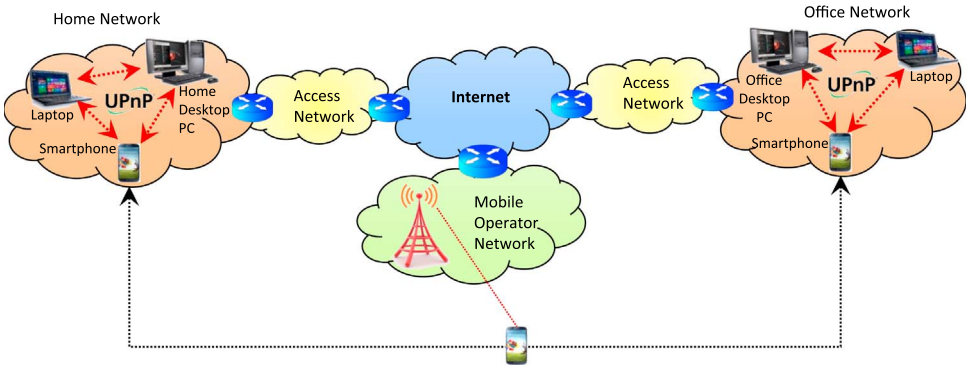



**Fig. 1.** The Ad-Hoc communication scenario in proposed system. The user has four different devices: a fixed home desktop PC, a fixed office desktop PC and two mobile devices (laptop and smartphone). The devices communicate directly with one another and run applications on only and only one device at a given moment based on priorities or user-specified configurations.

mobile devices. Instead of proxying applications, devices coordinate intelligently in proposed system to decide which one has to run the applications at a given moment. For example, when the smartphone is running applications, the desktop computer can sleep during that time. If the desktop computer is running applications, the smartphone can improve its battery life by reducing the utilization of its hardware components and switching OFF the applications.

### 4.2. Devices and applications control logics

Our Proposed system autonomously adapts to the environment and controls applications running on different user devices. Some possible control logics are the following:

#### 4.2.1. Based on device priority

The proposed system can control applications based on device priority. E.g., a desktop computer has higher priority than laptop, laptop has higher priority than tablet and tablet has higher priority than smartphone. The device priority may be based on weather the device is battery powered or not. When the higher priority device switches ON, our system will run applications on it and will stop on all other low priority devices.

#### 4.2.2. Based on pre-specified time periods

The proposed system can control applications based on the pre-specified time periods for each device. E.g., a user can instruct our system to run applications on office computer from 8:00 a.m. till 6:00 p.m., on smartphone from 6:00 p.m. till 8:00 p.m. and on home computer from 8:00 p.m. till 10:00 p.m. and don't run applications on any of his device from 10:00 p.m. till 8:00 a.m. (during sleeping time).

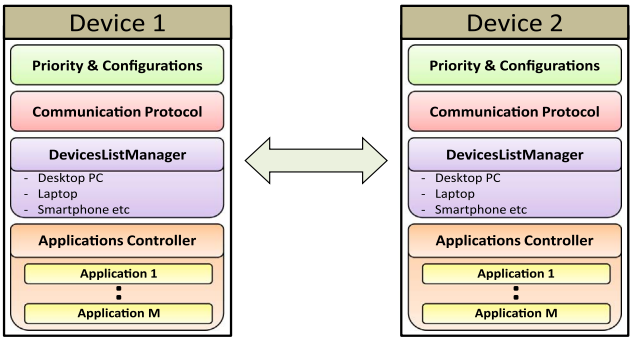



**Fig. 2.** The generic software architecture in proposed system.

#### 4.2.3. Mixed User Configurations

The proposed system can control applications on different user devices based on the mixed configurations. A user may want to run some of his applications on computer while some on smartphone based on application importance at the moment, e.g., never run facebook messenger on computer as browser is sufficient or never run skype on smartphone as he wants not to be disturbed outside work hours, etc. For example, a user has 5 different applications A, B, C, D and E. He wants his home computer to run all of them, office computer to run only applications A, B and E, and smartphone to run only applications C and D.

### 4.3. Reference architectural design

The proposed system is very flexible and operates in Ad-Hoc fashion. The devices communicate directly with one another without relying on any central coordinating unit. This design is very suitable for local networks which easily allow two-way communication among devices without NAT issues or compromising on Firewall. The user will have satisfactory concerns about the privacy and security of data as no third-party is involved in the scenario. Further, several communication paradigms (e.g., UPnP, multicast DNS) are available which allow devices to autonomously discover and seamlessly communicate with one another.

Fig. 1 presents reference architectural design considering a single user having four different devices: a fixed home desktop PC, a fixed office desktop PC and two mobile devices (laptop and smartphone). The communication among devices always takes place inside the local network while outside local network, the smartphone is responsible to maintain the presence of applications over 3G/4G data connection. In this Ah-Hoc scenario, it is important for all devices to connect with the same switch or WiFi access point so to be able to communicate. The desktop computers and laptops always stay connected to the local network, however, the smartphone needs to switch from 3G/4G data connection to WiFi whenever it enters in the local network. Fortunately, modern smartphones have readily attained this requirement. Due to comparatively low power consumption of WiFi over 3G/4G, the smartphones automatically switches to WiFi network whenever an active WiFi access point is discovered. Thus, the process is automatic and seamless without requiring configuration changes or any input from the user.

As is obvious from Fig. 1, a device needs to keep track of all available user devices in the local network, their priorities or user specified configurations. Thus, the devices mutually decide which one has to run the applications. The smartphone is responsible to run applications only when no other active device is available over WiFi connection or when it is using 3G/4G connection. The desktop computers or laptops also run a power saving module that keeps track

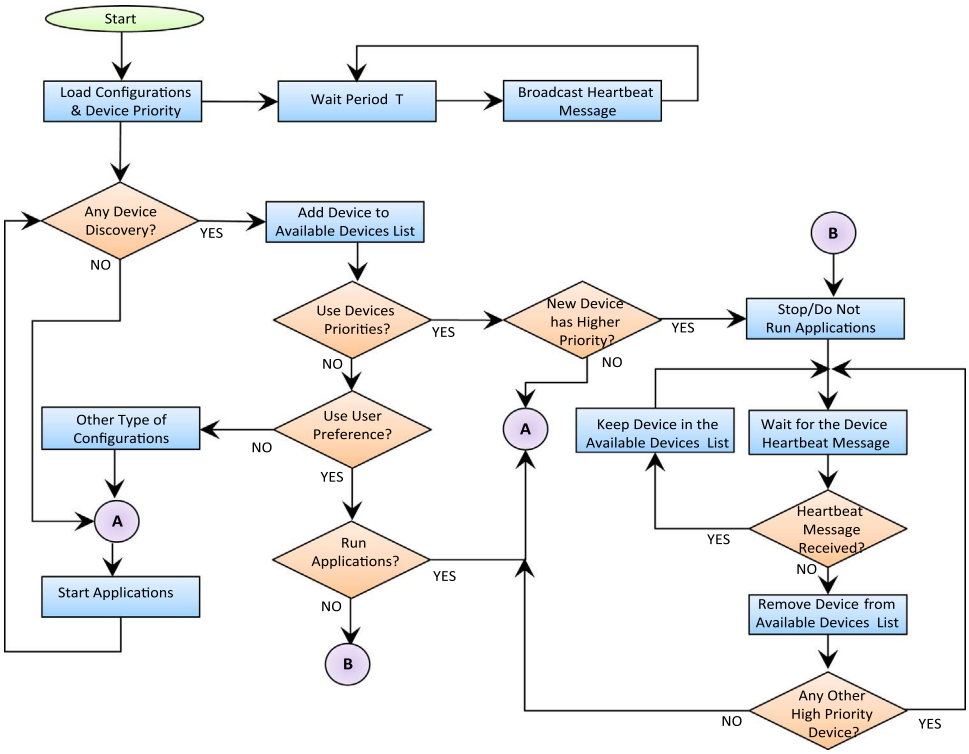



**Fig. 3.** A device functional flow chart in proposed system.

of power state changes from the kernel and immediately informs other available devices in the local network. Thus, once a high priority computer enters into sleep state, the other available devices again mutually decide about the next device to run the applications. A very simple and light-weight HGP is used to wake-up the desktop computer or laptop in case of remote access connection requests. It is likely possible that devices of different users may exist in the same network. Thus, to isolate devices of one user from devices of other users, a 128 bit UUID is assigned to each device for identification purpose. A whitelist of UUIDs is specified in each device configuration file based on which it ignores or processes requests from other devices. Note, identification based on IP addresses is not a good option as devices may be using DHCP.

Fig. 2 depicts the generic architectural design for devices in proposed system. It is obvious in Ad-Hoc design that just one software is needed that will run on all of the user devices. Each device needs to keep record of other available devices and their priorities or user specified configurations. A communication protocol is required for each device to discover and communicate with one another. Finally, the Applications Controller is responsible to run or stop applications based on the mutual decision among devices.

To have better understanding of each device functionality, Fig. 3 depicts a generic functional flow chart. Once a device powers up or enters in the local network, it loads the user defined priorities or configurations. The first step is to discover available devices in the local network. It will start running applications if no other device is available. In parallel, the device will also broadcast periodic presence or heartbeat messages in the local network. Once any other device is discovered, the software adds the device to available devices list and also checks if the applications control logic is requested based on priorities, user preference or any other type of mixed configurations. If other discovered device needs to run applications based on the control logic, the software will stop executing applications. Meanwhile, the software will start checking the periodic presence or heartbeat messages from that device to know if it is still available in the local network. Once other discovered device leave the network, the software will delete it from the available devices list and run applications by itself or any other available device based on the applications control logic. Fig. 3 depicts a very basic functional flow chart. However, the software could be more complex by incorporating many different types of applications control logics.

### 4.4. Communication framework

The proposed system also requires a communication framework that enables the devices to dynamically discover and communicate with one another. Thus, the devices can dynamically control one another based on the applications control logics. The communication framework in our proposed system is based on UPnP technology. The UPnP enables devices to automatically discover and seamlessly communicate with each other without requiring any configurations or user input. As soon as a device joins the network, it advertises itself and initiates search for other UPnP devices of interest in the network. The UPnP technology plays very important role to enable Ad-Hoc communication between devices as shown in Fig. 1. Each UPnP device consists of a description file in XML format that contains complete information about device including UUID, device type, model name and number etc. These parameters help identify if the other discovered device is a desktop PC or laptop or smartphone. More description about UPnP functionalities and implementation details are provided in Section 6.4.

### 4.5. Pros and cons of proposed system

This section summarizes pros and cons of proposed system compared to the original NCP concept (that is responsible to proxy also applications presence). Some of the pros include:

1. Instead of proxying, proposed system controls/schedules execution of applications on different user devices. Thus, the NCP issues

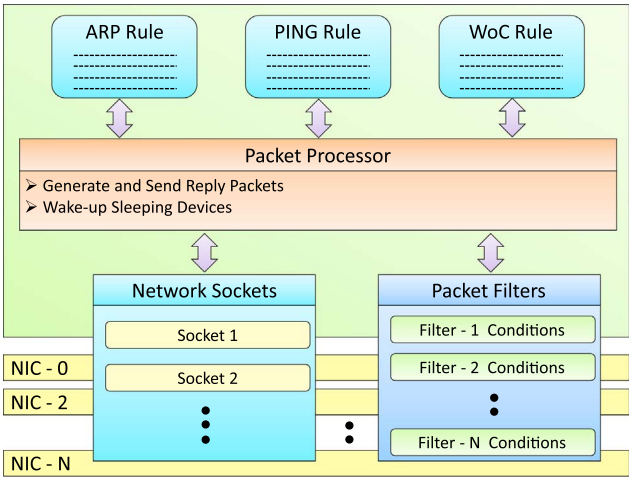



**Fig. 4.** The basic architectural design of HGP.

related to proxying of proprietary closed-source applications do not exist in the proposed system.
2. Since applications are not proxied in proposed system, no application specific privacy and security concerns exist. Note, the NCP concept requires important application secrets or privacy data in order to proxy the presence of a specific user account.
3. Since the proposed system does not proxy applications, there is no need to proxy their associated TCP sessions. Note, preserving TCP sessions on behalf of sleeping devices is a challenging task for NCP.
4. Proposed system uses a light-weight HGP (addressed in Section 5) that impersonates IP addresses of client devices. However, the scope is only limited to one local network and IP address collision issues do not exist. The NCP concept may face IP address collision issues if covering for more than one local networks.
5. There is no need to keep track of device mobility and its changing IP address in proposed system which is important in the NCP concept for mobile devices.

Some cons of proposed system include:

1. In terms of reducing the entire network energy waste, the proposed system provides energy savings equivalent or higher compared to NCP concept for fixed network devices. However, in terms of improving the battery life of smartphones, the benefits may be slightly lower than the NCP concept.
2. The proposed system solves the application proxying issues which exist in the original NCP concept. However, it cannot address the wake-up of sleeping computers on remote access connection requests. Thus, a very light and simple HGP is also required to achieve this objective. The design and capabilities of HGP is presented in Section 5.

## 5. Home Gateway Proxy

There are two important reasons due to which people never switch OFF their computers in offices and homes: (i) Internet connectivity for remote access and (ii) Internet connectivity for applications. Our proposed system only tackles with the second issue 'Internet connectivity for applications'. To allow remote access to sleeping devices, we take benefit from a very light-weight HGP. The HGP is responsible to impersonate the sleeping device IP address and to wake it up whenever a new connection request is received at a given transport protocol and port (e.g., TCP:22 for SSH, TCP:3389 or UDP:3389 for remote desktop protocol). The HGP is a subset of NCP [5] but with very limited practically realizable functionalities. Since, HEG runs on always ON gateway device with very low power consumption, it does not cause any incremental energy waste.

### 5.1. Basic architectural design

Fig. 4 depicts generic conceptual architecture for the HGP. It is somehow similar to NCP [5] and consists of four important components: (i) network traffic filters, (ii) network sockets, (iii) packet processor and (iv) a set of proxying rules. The proxying rules specify the operations that HGP needs to undertake on behalf of sleeping devices when a specific packet is received. Thus, each rule is implemented for a specific proxiable protocol and action. The job of the packet filters is to hijack and inspect packets intended for sleeping devices based on the conditions specified in each proxying rule. After a packet intended for sleeping device is received, it is passed to the packet processor block which then performs certain actions. Network sockets are used sending and receiving packets, e.g., sending wake-up packets, proxying ARP and PING, UPnP based communication with client devices, etc.

### 5.2. Proxying rules

The original NCP concept is quite complex that preserves open TCP connections and impersonates the applications running on client devices [1,5]. However, to achieve our objectives, we need a very basic set of functionalities. The HGP implements a subset of NCP rules. At present, the following rules are supported by HGP.

#### 5.2.1. ARP rule

The ARP rule generates and responds to ARP request packets on behalf of sleeping devices. It associates the HGP's (gateway device in our case) MAC address with the IP address of sleeping device. This rule is quite important as the final delivery of packet inside the local network is based on MAC address.

#### 5.2.2. PING rule

The PING rule generates and responds to ICMP echo-request packets on behalf of sleeping devices. In short, it maintains the presence of sleeping device IP address.

#### 5.2.3. Wake-on-connection rule

The Wake-on-Connection (WoC) rule simply wakes-up a sleeping device when a new connection request is received at a given transport protocol and port number e.g., TCP:22 for SSH, TCP:3389 or UDP:3389 for remote desktop protocol. This rule was originally specified in the NCP architecture [5] and is suitable for both connection oriented protocols (i.e., TCP) and connectionless protocols (i.e., UDP). This rule plays the main role in establishing a remote access connection with the sleeping device.

### 5.3. Communication framework

The HGP communicates with fixed network devices such as home and office desktop PCs. Note, the HGP is designed to offer services to more than one user and devices identification is based on UUIDs. The desktop PC implements a kernel module which tracks changes in its power state. These power state changes are communicated to HGP using UPnP communication framework. The UPnP also enables client devices to register ARP, PING and WoC rules at the HGP. The UPnP architecture can be most suitable choice for communication protocol to enable auto-discovery, zero configuration and seamless communication between the HGP and client devices (i.e., desktop PCs).

## 6. Implementations

The implementation of proposed system is based on the basic principles of versatility and extensibility. Due to Ad-Hoc nature, devices autonomously and seamlessly communicate with each other without requiring any configurations or input from the user. Thus, the

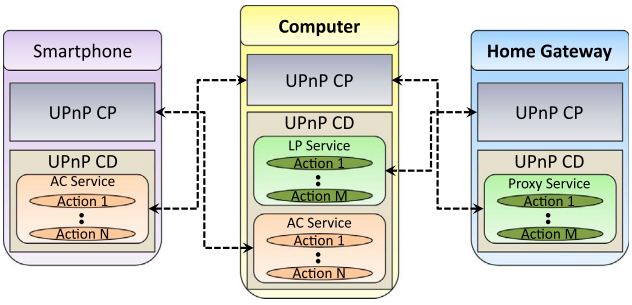



**Fig. 5.** UPnP communication framework design.

user can freely move around and/or switch ON/OFF devices without caring about transferring of applications control from one device to the other. All the operations in proposed system take place smoothly and seamlessly.

It can be observed from Sections 4 and 5 that the proposed system consists of: (i) intelligent Ad-Hoc collaboration among devices, and (ii) the HGP. A single piece of software is implemented for intelligent Ad-Hoc collaboration among devices and is executed on each user device. However, the HGP software is intended to be executed on gateway devices and its client needs to be executed on only computers (note, the HGP client is only for computers). At present, the HGP client is integrated in same Ad-Hoc collaboration software for computers. Thus, the proposed system consists of total three different softwares: (i) smartphone's software, (ii) computer's software and (iii) HGP software.

### 6.1. Smartphone's software

It enables the smartphone to interact with the other available devices in the local network using a UPnP communication framework. Based on the applications control logic, it mutually decides with the available devices whether to run the applications. The smartphone's software is responsible to run applications whenever it is outside the local network or uses 3G/4G data connection. Thus, there is possibility of not running the applications only when the smartphone is connected to the home WiFi network. To reduce the energy consumption on smartphone, it starts the discovery mechanism for local devices only when connected to WiFi network.

### 6.2. Computer's software

The computer's software is responsible not just to interact with the locally available user devices but also with the HGP using a UPnP communication framework. Like the smartphone's software, it also starts or stops applications based on the mutual decision with other local devices. The computer's software also includes a power saving module that keep track of the power state transitions from the kernel. Whenever the computer switches ON or goes to sleep, it immediately informs the local available devices as well as the HGP. The power state tracking is quite important to let the HGP decide when to start or stop proxying for the device. Further, it can let another user device e.g., the smartphone about when to start or stop the applications.

### 6.3. HGP software

The HGP software is very flexible and scalable and can concurrently maintain presence for multiple devices. Thus, it keeps track of all registered devices and their operational status. When a device enters into sleep state, its set of proxying rules are activated. The specific rules are deactivated once the device wakes up. The activation of rules imply setting of proper filters to sniff the packets intended for sleeping device based on the rules specifications. The packet filtering in the HGP framework is based on the Packet CAPturing (PCAP) library. Once a received packet matches the filter specifications, it is passed to the packet processor block of HGP for processing and determining of appropriate action.

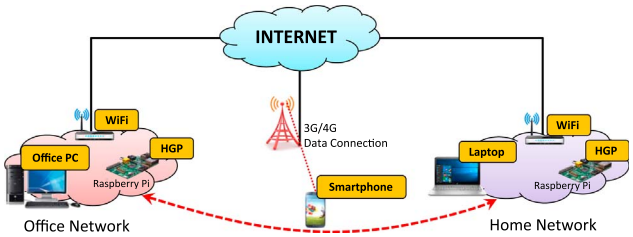



**Fig. 6.** Experimental testbed.

### 6.4. Communication framework

To achieve the objectives of zero-configuration and autonomous seamless communication among devices, we developed our communication framework based on the UPnP architecture [29]. The UPnP communication protocol provides a reliable and efficient exchange of information among user devices and the HGP. The UPnP Device Architecture (UDA) defined two main roles in the communication paradigm; the Controlled Device (CD) and the Control Point (CP). The CP (operates like a client) sends requests/commands to the CD (operates like a server) and receives back the response. The CD implements one or more services whereas each service supports a list of actions/commands. The UPnP CD manages complete description of the device and implemented services.

We have tailored the UPnP architecture to meet the needs of our communication framework as depicted in Fig. 5. It can be observed that each device implements a CP as well as a CD. Thus, each device can send and received commands from other devices. The CD on home gateway provides the proxy service which is responsible for proxying presence of sleeping devices, e.g., desktop PCs. The proxy service receives power state notifications from client devices based on which it activates or deactivates ARP, PING and WoC rules. The CD on smartphone provides an Application Controlling (AC) service. The AC service is responsible to start or stop applications based on the mutual decision with other user devices. The actions provided by AC service include SetPriority, UpdatePriority, AddApplication, RemoveApplication, StartApplications and StopApplications. The CD on computer implements two different types of services: AC and Low Power (LP). The LP service manages the operations related to device power state transitions and communicates with the HGP. The AC service receives commands from other user devices in the network and its functionalities are similar to the one running on smartphone.

### 6.5. Implementation tools

All three software entities were implemented in Linux operating system. Standard C/C++ was used as the main programming language. Several open source C/C++ libraries were also used for ease in implementations including libUPnP, PCAP, and BOOST libraries. The libUPnP libraries were used for implementation of UPnP communication framework. The libUPnP implementations are compatible with UPnP device architecture version 1.0 and supported by different OSes including embedded Linux and Microsoft Windows. The PCAP libraries were used for packet sniffing and filter purpose in HGP software. Packet filtering enables HGP to identify packets intended for sleeping devices. The BOOST libraries provide basic simplified functions and make easier the implementation of basic networking tasks. Most of the networking tasks in all three software entities were implemented with the help of BOOST libraries. The smartphone software was tested on a Ubuntu based smartphone, the HGP software on Raspbian OS and computer software was flexible and could run on both Windows and Linux OSes.

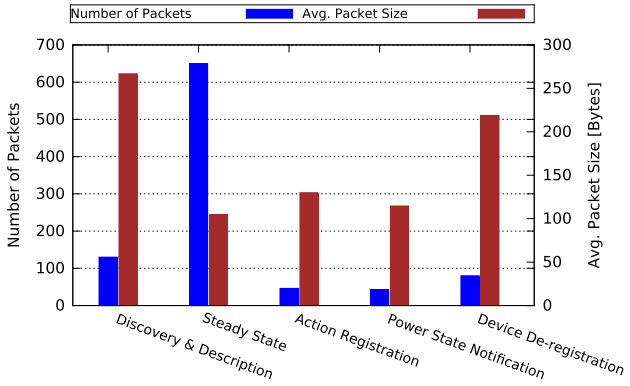



**Fig. 7.** Number of packets exchanged and average packet size during different UPnP events.

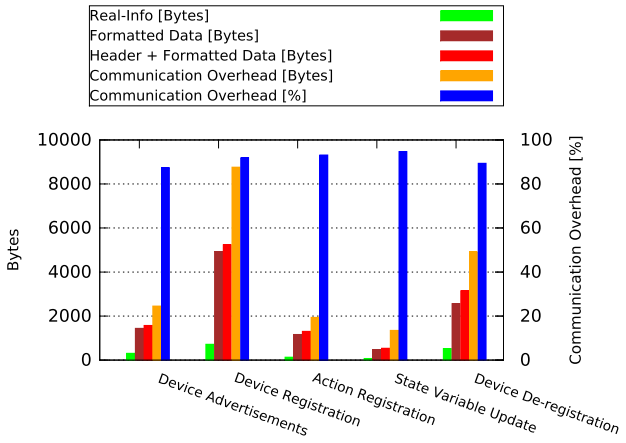



**Fig. 9.** UPnP overhead analysis in proposed framework.

## 7. Measurements and performance evaluation

In this section, we present the performance evaluation to verify the correctiveness of proposed scheme in realistic scenarios. The testbed used for experimental evaluation is shown in Fig. 6. The testbed consists of a desktop PC at office, a laptop at home and two HGPs running on low power Raspberry Pi pocket PCs in home and office environments. The scenario mainly involves communication among devices in home or office environments. Outside the home/office network, the smartphone is responsible to maintain presence of applications over 3G/4G data connection. The applications control logic is based on fewer devices (a typical user may use maximum 3 or 4 devices in his daily life). However, the HGP is notlimited to a single user but proxy presence of devices belonging to multiple users e.g., in office environment. Thus, the scalability of HGP can be a small concern as the devices in small/medium office usually never exceed over 100.

In the first phase of experimental evaluation, we verified the correct functioning of developed software prototypes including all proxying functionalities and features of the communication framework. The UPnP communication was quite robust and devices seamlessly discovered each other as soon as connected to the same network. The kernel module on laptop and desktop PC also worked efficiently in detecting transitions in power state and informing HGP immediately over UPnP. The developed Linux based application was also successfully tested on Ubuntu based smartphone. The devices were able to correctly execute applications based on the priorities of registered devices and user specified settings. As soon as smartphone enters in the office network, it automatically disconnects 3G/4G data connection and connects with WiFi. The UPnP discovery mechanism enables it to immediately register the desktop PC and stops (or keep running) applications based on defined user priorities.

In the second phase of evaluation, we analyzed the performance critical factors such as communication overhead, communication latencies and memory requirements.

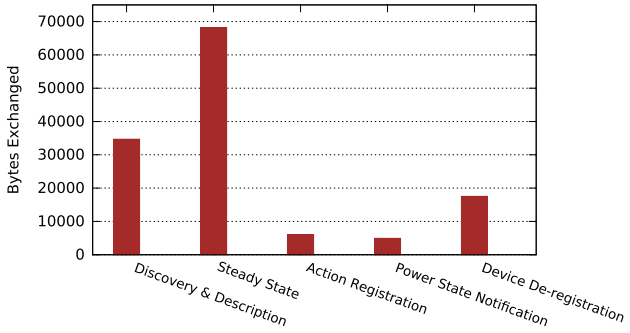



**Fig. 8.** Total Bytes exchanged during different UPnP events.

### 7.1. Communication overhead

One of the important design factor is communication overhead among devices. The protocol with high overhead could severely affect the real-time performance of application on low bandwidth channels. High overhead could significantly reduce the throughput on low bandwidth channels. For overhead analysis, we considered individual UPnP events including discovery and registration, steady-state condition, action registration, power state notification and device deregistration. We performed experiment over the duration of 12 mins during which a device registers with another device, registers an action, changes power state and notifies peer devices and finally de-registers with its peer device. Fig. 7 depicts the overhead in terms of total number of packets exchanged and average packet size during different events. Majority of the packets were very small in size and exchanged during steady state condition e.g., periodic advertisements. Packet are big during discovery, registration and de-registration phases due to carrying complete UPnP device and service descriptions. However, these messages are not very frequent. For more clarity, Fig. 8 depicts the total number of Bytes exchanged during different events which is directly linked with packet average size and frequency.

We also analyzed the UPnP based communication framework in terms of real information/data, overhead due to XML formatting of information, overhead due to headers and overhead due to UPnP communication paradigm. The real information represents the actual data that needs to be transferred from one device to the other (e.g., IP address, MAC address, device and service identifiers, state variables, etc). This actual information is then converted into XML format before being transmitted. The packet headers also create significant overhead. Most of the overhead is created by the UPnP communication paradigm by exchanging multiple packets during discovery, retrieval of device and service descriptions, etc. Fig. 9 depicts the more detailed classification of UPnP overhead. It can be observed that the overhead is quite high during the device registration phase as each device downloads the complete device and services descriptions of each other. The smallest overhead is during the notification of change in state variables, which in Fig. 9 corresponds to device power state variable.

### 7.2. Latencies analysis

Delay or latency is one of the critical factor that can affect the normal network operations. Thus, we measured the latency created due to UPnP communication and internal HGP processing during different rules registration. Table 1 summarizes our results averaged over 100 trials. It can be observed that the latencies are quite small and very stable for different rules, hence, will not leave any adverse effects on the network operations.

**Table 1**
Delay introduced during rules registration.

| | Min (ms) | Avg (ms) | Max (ms) | Std dev (ms) |
|---|---|---|---|---|
| ARP | 54.2 | 55.772 | 57.448 | 0.496 |
| PING | 53.7 | 55.91 | 57.754 | 0.728 |
| WoC | 54.452 | 56.213 | 58.043 | 0.822 |

**Table 2**
Round Trip Time measured on PING traffic with and without the intervention of HGP. The results are averaged over 100 samples.

| | Min (ms) | Avg (ms) | Max (ms) | Mean Dev (ms) |
|---|---|---|---|---|
| Host Awake | 0.433 | 0.462 | 0.528 | 0.020 |
| Host Sleeping | 1.347 | 1.651 | 1.834 | 0.179 |

Another important consideration is the latency experienced by third-party devices when the HGP responds on behalf of sleeping devices. We measure the latency for PING request packets with and without the intervention of HGP as shown in Table 2. We observed that the packet loss was zero regardless of the frequency of PING requests. However, the latency or Round Trip Time (RTT) is quite large when the HGP responds on behalf of sleeping devices. This latency is due to sniffing of PING requests and generation and transmission of PING response packets by the HGP. However, the values are still quite small and will not cause any adverse effects on network operations.

### 7.3. Memory requirements

We analyzed the resource requirements in terms of CPU and memory for all developed software prototypes. We observed that the CPU usage by developed software prototypes is always less than 5% due to no CPU intensive functionality involved in proposed system. User devices such as desktop PCs, laptops and smartphones are equipped with large amount of memory. However, the legacy gateway devices (i.e., to run the HGP) are normally equipped with 16 MB or 32 MB of memory. Thus, the memory requirement of HGP is critical to perform its functionalities. Table 3 reports the memory requirements by developed software prototypes. It can be observed that the memory requirement is very low in proposed system. Since, the HGP is expected to cover for large number of devices especially in the office environment, the memory requirement per additional registered client device must be very low. Table 4 presents the memory requirement for covering every additional client device. Due to very low memory requirement, the HGP with 8 MB of free memory can successfully proxy roughly 450 client devices.

### 7.4. Expected energy savings

The proposed system provides two types of benefits: (i) reducing entire network energy waste by allowing unattended desktop computers/laptops to sleep while maintaining applications presence on smartphones and (ii) improving battery life of smartphones by not running the applications 27/4. The expected energy savings depend on the amount of time devices stay in the sleep state. This typically depends on the user behavior as well as the capabilities of HGP and applications state maintenance on smartphones. The smartphone is a low power always-ON device which normally run applications 24/7. In the proposed system, the smartphone runs applications for the shortest possible time (when no other user device is active), thus, improves battery life through reduced utilization of hardware components such as CPU, memory, WiFi, 3G/4G or other built-in sensors.

**Table 3**
Memory requirements by developed software entities.

| Computer | Smartphone | HGP |
|---|---|---|
| 3.29 MB | 1.816 MB | 2.67 MB |

**Table 4**
Memory requirement by HGP for every additional client. The results are averaged over 100 clients.

| Min (KB) | Avg (KB) | Max (KB) | Mean Dev (KB) |
|---|---|---|---|
| 14.93 | 18.071 | 21.143 | 0.869 |

**Table 5**
Daily idle period for devices in home and office environments [3].

| | Desktop PCs | | Laptops | |
|---|---|---|---|---|
| | Office | Home | Office | Home |
| Idle Period (%age) | 59.1 | 17.6 | 18.5 | 9.5 |
| Idle Period (hours) | 14.184 | 4.224 | 4.44 | 2.28 |

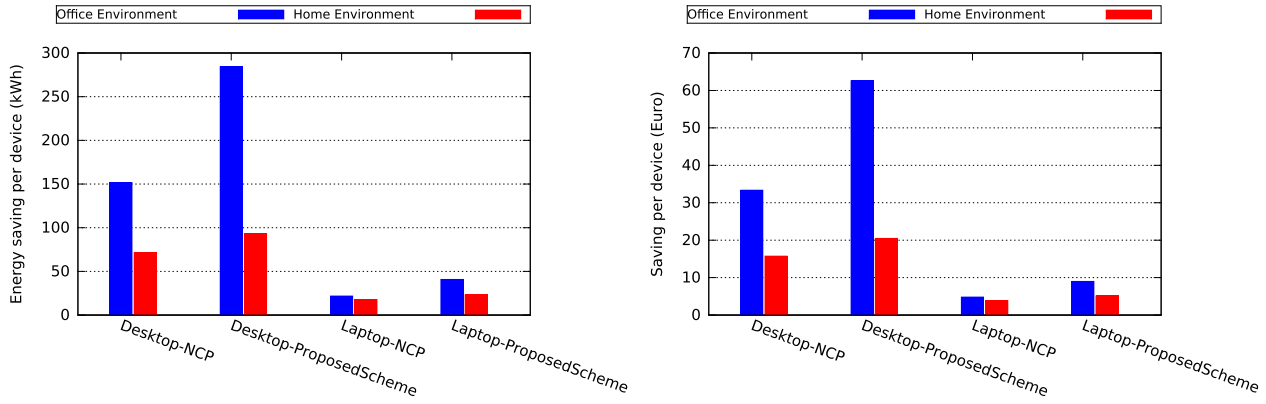
Based on reported results in [14], the applications on a typical smartphone consumes about 30% of its battery on a full charge. While the rest of the battery is consumed by display and operating system during idle state. Our proposed system has the potential to reduce battery usage by applications to 10% by running applications only when desirable. This means that a typical smartphone with 24 hours as battery discharge time on full charge, can run up to 30.8 hours by adaptation of our proposed scheme. This means that the proposed system has the potential to increase smartphone battery life by at-least 23%.

The authors in [3] performed detailed network traffic analysis in typical home and office environments. Based on the network traffic analysis, Table 5 reports the idle time for a desktop PC and laptop in home and office environments. The NCP has only Basic Proxying (BP) capabilities, i.e., proxying ARP, PING, DHCP, etc and cannot proxy proprietary closed source applications. Thus, a client device in NCP concept can sleep for 48% and 76% of the idle time in office and home environments, respectively. If the applications presence is also maintained (i.e., our proposed system with Full Proxying (FP) capabilities) then a device can sleep for 90% and 99% of idle time in office and home environments, respectively. Table 6 reports the sleep duration for desktop PCs and laptops with NCP concept (i.e., BP) and with our proposed system (i.e., FP).

To estimate the energy savings, we assumed 65 W and 4 W power consumption of a typical desktop PC in idle state and sleep state, respectively. For a typical laptop, we assumed 30 W and 2 W power consumption in idle state and sleep state, respectively. Fig. 10 depicts the expected energy savings for a single device. It is estimated that our proposed system provides approximately 46% and 23% higher energy savings compared to the NCP concept in office and home environments, respectively. This is due to the fact that NCP lacks the ability to proxy proprietary closed source applications. Considering 22 cents/kWh as the average cost of electricity in Europe, up to 60 Euro annual

**Table 6**
Annual sleep duration for devices in home and office environments.

| | Office | | Home | |
|---|---|---|---|---|
| | BP | FP | BP | FP |
| Sleep (%age of idle time) | 48% | 90% | 76% | 99% |
| Desktop Sleep/day (hours) | 6.81 | 12.77 | 3.21 | 4.18 |
| Laptop Sleep/day (hours) | 2.13 | 3.99 | 1.733 | 2.26 |
| Desktop Sleep/year (hours) | 2485.65 | 4661.05 | 1171.65 | 1525.7 |
| Laptop Sleep/year (hours) | 777.45 | 1456.35 | 632.545 | 824.9 |




(a) Savings in kWh (b) Savings in Euro

**Fig. 10.** Expected annual savings per device: (a) Savings in kWh, (b) Savings in Euro.

savings per device can be achieved. Considering millions of Internet connected devices in the world, the proposed scheme has potential to provide annual savings in billions if adopted worldwide.

## 8. Conclusion

The original NCP concept is very useful in terms of reducing the network energy waste. However, it faces many critical issues and challenges especially related to preserving open TCP connections and proxying proprietary closed-source applications. Thus, there is a strong need for a complementary approach that can overcome the limitation of NCP concept.

Today, an Internet-connected smartphone is becoming an important part of our daily life and playing a role in reducing the network energy waste. The modern smartphones can run the same operating systems and applications as the desktop computers. Thus, in this paper, we presented an intelligent collaborative scheme among daily used fixed and mobile devices that guarantees to run applications on one and only one user device at any given moment. The Ad-Hoc nature of proposed design provides more communication privacy among devices belonging to a single user. We also presented the implementation instructions for proposed system and highlighted the embedded features such as zero-configuration, auto-discovery and autonomous seamless communication, etc.

To allow remote access to sleeping devices, we deployed a HGP with a very basic set of practically realizable features. The HGP software is only responsible to impersonate the sleeping device IP address and wake it up when a connection request at a given transport protocol and port is received. Our proposed system effectively overcomes the limitation of the original NCP concept in maintaining transport and application layers presence on behalf of sleeping devices. In short, our approach is practically realizable and can provide huge amount of energy savings annually.

## References

[1] R. Bolla, R. Khan, M. Repetto, Assessing the potential for saving energy by impersonating idle networked devices, IEEE J. Sel. Areas Commun. 34 (5) (2016) 1676–1689. http://dx.doi.org/10.1109/JSAC.2016.2545414.

[2] K.J. Christensen, C. Gunaratne, B. Nordman, A.D. George, The next frontier for communications networks: power management, Comput. Commun. 27 (18) (2004) 1758–1770. http://dx.doi.org/10.1016/j.comcom.2004.06.012.

[3] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, N. Taft, Skilled in the art of being idle: reducing energy waste in networked systems, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association: Berkeley, CA, USA, 2009, pp. 381–394.

[4] R. Khan, S.U. Khan, Achieving energy saving through proxying applications on behalf of idle devices, Procedia Computer Science, 83 (2016) pp. 187–194, In: Proceedings of the 7th International Conference on Ambient Systems, Networksand Technologies (ANT 2016)/The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016)/Affiliated Workshops. ⟨http://dx.doi.org/http://dx.doi.org/10.1016/j.procs.2016.04.115⟩.

[5] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Network connectivity proxy: architecture, implementation and performance analysis, IEEE Syst. J. PP (99) (2015) 1–12. http://dx.doi.org/10.1109/JSYST.2015.2438639.

[6] M. Jimeno, K. Christensen, B. Nordman, A network connection proxy to enable hosts to sleep and save energy, in: 2008 IEEE International Performance, Computing and Communications Conference, 2008, pp. 101–110. ⟨http://dx.doi.org/10.1109/PCCC.2008.4745133⟩.

[7] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Network connectivity proxy: an optimal strategy for reducing energy waste in network edge devices, in: 2013 Proceedings of the 24th Tyrrhenian International Workshop on Digital Communications - Green ICT (TIWDC), 2013, pp. 1–6. ⟨http://dx.doi.org/10.1109/TIWDC.2013.6664214⟩.

[8] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, R. Gupta, Somniloquy: augmenting network interfaces to reduce PC energy usage, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 365–380.

[9] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Design and implementation of cooperative network connectivity proxy using universal plug and play, in: The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons, 2013, pp. 323–335.

[10] R. Khan, R. Bolla, M. Repetto, R. Bruschi, M. Giribaldi, Smart proxying for reducing network energy consumption, in: 2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012, pp. 1–8.

[11] P. Werstein, W. Vossen, A low-power proxy to allow unattended jabber clients to sleep, in: 2008 Proceedings of the Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, pp. 390–395. ⟨http://dx.doi.org/10.1109/PDCAT.2008.18⟩.

[12] R.B. Jennings, E.M. Nahum, D.P. Olshefski, D. Saha, Z.-Y. Shae, C. Waters, A study of internet instant messaging and chat protocols, IEEE Netw. 20 (4) (2006) 16–21. http://dx.doi.org/10.1109/MNET.2006.1668399.

[13] R. Bolla, M. Chiappero, R. Khan, M. Repetto, Saving energy by delegating network activity to home gateways, IEEE Trans. Consum. Electron. 61 (4) (2015) 445–453. http://dx.doi.org/10.1109/TCE.2015.7389798.

[14] R. Bolla, R. Khan, X. Parra, M. Repetto, Improving smartphones battery life by reducing energy waste of background applications, in: 2014 Proceedings of the Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, 2014, pp. 123–130. ⟨http://dx.doi.org/10.1109/NGMAST.2014.10⟩.

[15] J. Klamra, M. Olsson, K. Christensen, B. Nordman, Design and implementation of a power management proxy for universal plug and play, in: Proceedings of SNCW, 2005.

[16] M. Jimeno, K. Christensen, A prototype power management proxy for gnutella peer-to-peer file sharing, in: Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN 2007), 2007, pp. 210–212. ⟨http://dx.doi.org/10.1109/LCN.2007.93⟩.

[17] L. Irish, K.J. Christensen, A green TCP/IP to reduce electricity consumed by computers, in: Southeastcon '98. Proceedings. IEEE, 1998, pp. 302–305. ⟨http://dx.doi.org/10.1109/SECON.1998.673356⟩.

[18] C. Gunaratne, K. Christensen, B. Nordman, Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed, Int. J. Netw. Manag. 15 (5) (2005) 297–310. http://dx.doi.org/10.1002/nem.565.

[19] Y. Agarwal, S. Savage, R. Gupta, SleepServer: a software-only approach for reducing the energy consumption of PCs within enterprise environments, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, USENIX Association: Berkeley, CA, USA, 2010, pp. 22–22.

[20] F.R. Dogar, P. Steenkiste, K. Papagiannaki, Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, ACM: New York, NY, USA, 2010, pp. 107–122. ⟨http://dx.doi.org/10.1145/1814433.1814446⟩.

[21] M. Anand, E.B. Nightingale, J. Flinn, Self-tuning wireless network power management, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom '03, ACM: New York, NY, USA, 2003, pp. 176–189. ⟨http://dx.doi.org/10.1145/938985.939004⟩.
[22] J. Hare, D. Agrawal, A. Mishra, S. Banerjee, A. Akella, A network-assisted system for energy efficiency in mobile devices, in: 2011 Proceedings of the Third International Conference on Communication Systems and Networks (COMSNETS 2011), 2011, pp. 1–10. ⟨http://dx.doi.org/10.1109/COMSNETS.2011.5716500⟩.
[23] M. Gundlach, S. Doster, H. Yan, D.K. Lowenthal, S.A. Watterson, S. Chandra, Dynamic, power-aware scheduling for mobile clients using a transparent proxy, in: Parallel Processing, 2004. ICPP 2004. International Conference on, vol.1, 2004, pp. 557–565. ⟨http://dx.doi.org/10.1109/ICPP.2004.1327966⟩.
[24] V. Looga, Y. Xiao, Z. Ou, A. Yl-Jski, Exploiting traffic scheduling mechanisms to reduce transmission cost on mobile devices, in: 2012 IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 1766–1770. ⟨http://dx.doi.org/10.1109/WCNC.2012.6214070⟩.
[25] I. Kelnyi, J.K. Ludnyi, Nurminen, Using home routers as proxies for energy-efficient bittorrent downloads to mobile phones, IEEE Commun. Mag. 49 (6) (2011) 142–147. http://dx.doi.org/10.1109/MCOM.2011.5783999.
[26] J. Kim, H. Joe, H. Kim, O-Sleep: Smartphones Output-Oriented Power Saving Mode, in: Computational Science and Engineering (CSE), 2012 IEEE Proceedings of the 15th International Conference on, 2012, pp. 334–340. ⟨http://dx.doi.org/10.1109/ICCSE.2012.53⟩.
[27] R. Bolla, M. Giribaldi, R. Khan, M. Repetto, Smart Proxying: An Optimal Strategy for Improving Battery Life of Mobile Devices, in: Green Computing Conference (IGCC), 2013 International, 2013, pp. 1–6. ⟨http://dx.doi.org/10.1109/IGCC.2013.6604478⟩.
[28] R. Khan, R. Bolla, M. Repetto, Design of UPnP based Cooperative Network Connectivity Proxy, in: Sustainable Internet and ICT for Sustainability (SustainIT), 2012, 2012, pp. 1–3.
[29] UPnP forum, 2012. URL: ⟨http://www.upnp.org⟩.